

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 05-150931

(43)Date of publication of application : 18.06.1993

(51)Int.Cl.

G06F 3/14
// G06F 9/44

(21)Application number : 03-076793

(71)Applicant : INTERNATL BUSINESS MACH CORP
<IBM>

(22)Date of filing : 16.03.1991

(72)Inventor : SHACKELFORD FLOYD W
MOORE RICHARD E

(30)Priority

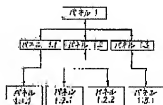
Priority number : 90 510350 Priority date : 17.04.1990 Priority country : US

(54) METHOD FOR PROCESSING SCOPE INSTRUCTION CHANGE AND DEVICE FOR THE SAME

(57)Abstract:

PURPOSE: To change a scope from an active panel without being suppressed by another panel which can be seen on a display screen by processing a scope change instruction prepared by a user by selection-controlling a panel hierarchized so that a successive attribute can be held by a hierarchical inside program logic.

CONSTITUTION: When a panel 1, 2 receives a scope changing instruction related with a phenomenon which cannot be processed by itself, the panel 1, 2 transfer the information to a slave panel 1, 2, 1. The panel 1, 2, 1 does not have a slave panel which can receive the changing instruction, and the panel 1, 2, 1 returns unchanged data and the scope instruction to the master panel. The panel 1, 2 transmits the changing instruction to the other slave panel of the panel 1, 2. A panel 1, 2, 2 makes a response in the same way as the panel 1, 2, 1. At this point, the panel 1, 2 returns the scope changing instruction and the data to the master panel (panel 1). The panel 1 can process the scope changing instruction, and the panel 1 can be an active panel.



スコープ変更
指示
スコープ変更
指示

特開平5-150931

(43)公開日 平成5年(1993)6月18日

(51)Int.Cl.⁴G 0 6 F 3/14
// G 0 6 F 9/44

識別記号

3 5 0 C 7165-5B
3 3 0 Z 9193-5B

庁内整理番号

F I

技術表示箇所

審査請求 有 請求項の数17(全 15 頁)

(21)出願番号 特願平3-76793

(22)出願日 平成3年(1991)3月16日

(31)優先権主張番号 0 7 / 5 1 0 3 5 0

(32)優先日 1990年4月17日

(33)優先権主張国 米国 (U S)

(71)出願人 390009531

インターナショナル・ビジネス・マシーン
ズ・コーポレーションINTERNATIONAL BUSIN
ESS MACHINES CORPO
RATIONアメリカ合衆国10504、ニューヨーク州
アーモンク (番地なし)(72)発明者
フロイド・ウェイン・シャツクルフオード
アメリカ合衆国、ジョージア州30518、パ
フォード、ハノーバー・ドライブ 3510番
地

(74)代理人 弁理士 須宮 孝一 (外2名)

最終頁に続く

(54)【発明の名称】 スコープ変更指令処理方法及び装置

(57)【要約】

【目的】本発明は、アクティブパネルから表示画面上にて見ることができる他のパネルに抑止されることなくスコープを変更できるようにするパネル間プロセスフロー制御方法及び制御システムを提供する。

【構成】このプロセスはオブジェクト指向プログラミング構成に依存しており、特に継承属性及びそのパネルクラスに割り当てられている制御メソッドに依存している。各子パネルがその親パネルの制御メソッドを継承するようにした階層的内部プログラムロジックが実現される。アクションルータ制御メソッドは、現在のアクティブパネルがスコープ変更要求を局部的に処理できるか否かを確認する。現在のアクティブパネルの子パネル達は、そのスコープ変更要求を処理できる子パネルを発見できるまで、当該継承された制御メソッドを用いて制御された順序で照会される。

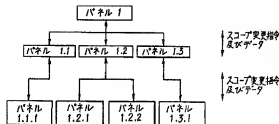


図5 パネル及び子パネル

【特許請求の範囲】

【請求項1】中央データプロセスと、ランダムアクセスメモリと、直接アクセス記憶装置と、表示用端末装置と、入力装置とを有する情報処理システムにおいて、現在アクティブな1つのパネルを含む複数のパネルを上記表示用端末装置に表示するようにしたオブジェクト指向環境によって、ユーザによって作られるスコープ変更指令を処理する方法において、

上記表示された複数のパネルのうちの1つを選択してスコープ変更指令の実行を開始することにより次のアクティブパネルにするステップと、

現在のアクティブパネルが上記スコープ変更指令を処理できるか否かを確認するステップと、

上記スコープ変更指令を処理できる非アクティブパネルを所定の順序で探索するステップと、

上記探索を通じて発見できた非アクティブパネルを上記表示用端末装置の前景にもつてくることにより新しいアクティブパネルとして指定するステップとを含むことを特徴とするスコープ変更指令処理方法。

【請求項2】上記スコープ変更指令を処理できる非アクティブパネルを所定の順序で探索する上記ステップは、上記スコープ変更指令を処理できる上記非アクティブパネルを発見できるまで、上記現在のアクティブパネルの子パネルをそれぞれ照会するステップと、上記現在のアクティブパネルの子パネルのいずれも上記スコープ変更指令を処理できないとき、上記現在のアクティブパネルの親パネルに上記スコープ変更指令を渡すことによりさらに処理をするステップとを含むことを特徴とする特許請求の範囲第1項に記載のスコープ変更指令処理方法。

【請求項3】上記表示された複数のパネルの各パネルについてすべての子パネルに関する階層関係リストを保持するステップを含むことを特徴とする特許請求の範囲第1項に記載のスコープ変更指令処理方法。

【請求項4】次のアクティブパネルを選択する上記ステップは、

マウス入力装置を使用して上記表示用端末装置上のスクリーンカーソルを、現在のアクティブパネルから、次のアクティブパネルになるべき表示されたパネルの可視領域に移動させるステップを含むことを特徴とする特許請求の範囲第1項に記載のスコープ変更指令処理方法。

【請求項5】上記所定の順序は、

現在のアクティブパネルから、当該現在のアクティブパネルからそれぞれ出ている分岐を下つてスコープ変更指令を処理できる子パネルを発見できるまで引き続き進行するようにすべての子パネルの階層関係に従うことを特徴とする特許請求の範囲第3項に記載のスコープ変更指令処理方法。

【請求項6】上記表示された複数のパネルはそれぞれパネル識別タグを含み、各スコープ変更指令は上記パネル

識別タグに対する変更を含み、スコープ変更指令を処理できる非アクティブパネルを探索する上記ステップは、検査される各パネルの上記パネル識別タグを当該パネル識別タグに対する上記変更と比較することにより一致するか否かを確認するステップを含むことを特徴とする特許請求の範囲第2項に記載のスコープ変更指令処理方法。

【請求項7】所定の順序に従って探索する上記ステップは、

上記スコープ変更指令を受け取った親パネルによつて上記スコープ変更指令が処理されることによりさらに処理を続けることができるか否かを確認するステップと、それまでに照会されていなかった上記親パネルの子パネルをそれぞれ上記スコープ変更指令を処理できるパネルを発見できるまで照会するステップと、上記親パネルのいずれの子パネルも上記スコープ変更指令を処理できないとき上記スコープ変更指令を上記階層関係内の一段高いレベルの親パネルに渡すステップとを含むことを特徴とする特許請求の範囲第5項に記載のスコープ変更指令処理方法。

【請求項8】中央データプロセスと、ランダムアクセスメモリと、直接アクセス記憶装置と、表示用端末装置と、入力装置とを有し、上記表示用端末装置に複数のパネルを表示し、かつ1つのパネルを現在のアクティブパネルにするようになされた情報処理システム上を走るオブジェクト指向環境について、ユーザによって作られたスコープ変更指令を処理するシステムにおいて、階層関係にある表示されたパネル群を連結すると共に、調査順序を確立する連係リスト手段と、

上記表示された複数のパネルが上記入力装置からのスコープ変更指令を受け取ることができるようにする第1のロジック手段と、

上記関連リスト手段と協働して非アクティブパネルに対するスコープ変更指令の経路を所定の順序で制御する第2のロジック手段と、

上記第2のロジック手段により形成された経路にある非アクティブパネルのいずれかによつて上記スコープ変更指令を処理できるか否かを確認すると共に、適切なパネルを発見できたとき上記第2のロジック手段に肯定的応答を送る第3のロジック手段とを含むことを特徴とするスコープ変更指令処理装置。

【請求項9】上記連係リスト手段はランダムアクセスメモリ内に格納されているデータ構造であることを特徴とする特許請求の範囲第8項に記載のスコープ変更指令処理装置。

【請求項10】上記第1のロジック手段、上記第2のロジック手段及び上記第3のロジック手段はランダムアクセスメモリ内に格納されているオブジェクト指向プログラミングモジュールであることを特徴とする特許請求の範囲第8項に記載のスコープ変更指令処理装置。

【請求項 11】 スコープ変更指令を生成するために使用される上記入力装置は、上記表示用端末装置上のスクリーンカーソルの位置を制御するマウスであることを特徴とする特許請求の範囲第 8 項に記載のスコープ変更指令処理装置。

【請求項 12】 上記表示された複数のパネル間の階層的親子関係に従う上記所定の順序は、先ず現在のアクティブパネルから出ている各分岐を上記すべての分岐を探索し終るまで下り続け、次に現在のアクティブパネルの親パネルに昇ることを特徴とする特許請求の範囲第 8 項に記載のスコープ変更指令処理装置。

【請求項 13】 上記第 1 のロジック手段、上記第 2 のロジック手段及び上記第 3 のロジック手段は「親」クラスと呼ばれるオブジェクト指向クラスをカプセル化する制御メソッドを含み、

上記オブジェクト指向クラスを表示されたパネル間の階層関係に基づいて各子パネルによって当該親パネルから継承することを特徴とする特許請求の範囲第 10 項に記載のスコープ変更指令処理装置。

【請求項 14】 表示画面上の複数のパネルを有する並行アプリケーション群との対話に基づいて、アクティブパネルの変更に関するユーザ要求を処理する方法であつて、

上記複数のパネルのうちの 1 つを選択して次のアクティブパネルとするステップと、
選択された次のアクティブパネルと対話する上記並行アプリケーションを所定の順序で探索するステップと、
選択された次のアクティブパネルを上記表示画面上の前景にもつくるステップとを含むことを特徴とするユーザ要求処理方法。

【請求項 15】 上記複数のパネルのうちの 1 つを選択する上記ステップは、現在のアクティブパネルから上記次のアクティブパネルに表示画面カーソルを移動させるマウス入力装置の使用を含むことを特徴とする特許請求の範囲第 14 項に記載のユーザ要求処理方法。

【請求項 16】 上記並行アプリケーションを所定の順序で探索する上記ステップは、階層的親子関係にある上記複数のパネルのリストを保持するステップと、
上記アクティブパネルから開始して階層的親子関係にあるパネルの上記リストの各分岐を下り続け、上記選択された次のアクティブパネルを発見できるまで遭遇する各パネルを検査するステップと、
上記選択された次のアクティブパネルを発見できなかったとき、上記アクティブパネルの親パネルに戻って当該親パネルから下っている分岐のうち未だ検査されていないパネル分岐をそれぞれ下り続けて探索を継続することにより遭遇する各パネルを検査するステップとを含むことを特徴とする特許請求の範囲第 14 項に記載のユーザ

要求処理方法。

【請求項 17】 各パネルを検査する上記ステップは、上記複数のパネルにそれぞれ対応する固有のパネル識別子を、各ユーザ要求に対応するパネル識別子に対する固有の変更と比較して一致していることを表示するステップを含むことを特徴とする特許請求の範囲第 16 項に記載のユーザ要求処理方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明はスコープ変更指令処理方法及び装置に関し、特にオブジェクト指向プログラミングシステムについて同一のアプリケーションに多数のアクティビティが同時に発生するようなウィンドウ環境（これを「メッシーデスク (messy desk)」ウィンドウ環境と呼ぶ）において、プログラム及び手続きの階層的制御機構を実現するための方法及びシステムに関する。

【0002】

【従来の技術】 パネル及びウィンドウはインタフェース要素を提示するために任意に入れかえながら使用され、インタフェース要素は表示画面においてユーザに対してオブジェクト及びアクションを提供する。例えば、「IBM コンピュータ PS/2」においてオペレーティングシステム「OS/2」により提供されるマルチタスク機能は、多数のアプリケーションが、「OS/2」のプレゼンテーション管理要素を用いて端末のモニタ画面を共用できるようにする。ユーザが現在対話しているウィンドウ（及びパネル）を「アクティブウィンドウ（及びアクティブパネル）」と呼ぶ。

【0003】 ユーザ環境における主要な可視要素は、背景画面、ウィンドウ（及びパネル）、アプリケーションアイコン、自由可動のカーソルである。点指選択概念はユーザ制御インタフェースにおいては基本的なものであり、ユーザは画面上の可視点にカーソルを移動させた後その点を選択する。現在選択されているパネル又はウィンドウの領域を超えてカーソルが移動されると、参照可能範囲すなわちスコープに変化が生ずる。「メッシーデスク」ウィンドウ環境においてユーザは、全く任意かつ予告なしに、パネル単位でスコープを変更することができる。他のパネルを選択すると当該選択されたパネルが前景に出されてアクティブパネルとなる。新しく選択したパネルをアクティブパネルにする基礎的制御機構はこれをユーザが目指すことはできない。見るように構造化されていないこの手続きの流れを制御するためには内部プログラムロジックが必要である。多くの「メッシーデスク」ウィンドウアプリケーションは単一マスタ手法を使用し、これにより画面上のすべてのパネルについて単一のマスタリストを保持するようになされている。従ってアプリケーションは選択すべきパネルを決定するために、各パネルの特性に関して十分に精進していなければならない。そうでない場合、アプリケーション

は、ユーザの入力を受け入れようとしているパネルを発見できるまで各パネルの間合わせを続ける必要がある。この手法は、各スコープ変更指令に関して、平均的に見てパネルのリストを半分程度は走査する結果になる。このような従来のリストを用いる手法とは異なり、本発明は、一方では完全に非抑止的なスコープ変更をなし得るようにすると同時に、プログラム及び手続きの階層的制御を実現するプロセスを提供する。

【0004】従来技術として、1987年7月に刊行されたIBM技術開示報告(IBM Technical Disclosure Bulletin)第30巻、736～738頁及び米国特許第4,845,644号に、パネルの連係リスト表現の実施例が示されている。上記IBM技術開示報告においては、順序付けに基づいて参照される経路順に従ってパネルが表示される。パネル間の経路を特定するためパネル生成時に接続行列を使用し、これによりアプリケーションプログラムに論理構造をプログラムする必要をなくすようになされている。上記米国特許等は、パネル表示の優先順位を変更できるようにした方法を開示しているが、本質的には、最後に用いた順序付け体系に依存している。

【0005】コンピュータに基礎を置くシステムがますます複雑になるにつれて、オブジェクト指向プログラミングが一層注目され、盛んに研究された。オブジェクト指向プログラミングシステムにおける主な焦点は、機能よりデータにある。「オブジェクト」は、データ構造及び当該データ構造にアクセスできる一組の操作又は機能により形成される。データ構造は、各スロットがデータの属性を含んでいる多数のスロットを含むフレームによって表現することができる。各属性としては、プリミティブ(すなわち整数又は文字列)、又は他のオブジェクトのインスタンス(個体)を指すオブジェクト参照を適用し得る。データ構造にアクセスすることができる各操作(機能)は「メソッド」と呼ばれる。通常、定義された各オブジェクトは一群のインスタンスに明示され、各インスタンスは特定のオブジェクトに対して特定のデータ構造を含んでいる。オブジェクト指向プログラミングシステムの主な特徴は、「カプセル化」及び「継承」である。フレームは当該フレームのメソッドによってカプセル内に密封される。そのことは、データ構造へのすべてのアクセスが、周囲を取り囲んでいるメソッドによって処理されることを意味しており、これによりデータの独立性が確保されることを意味している。継承特性はオブジェクトの新しいスーパークラス及びサブクラスを作成することにより、前に書かれたプログラムを拡張できるようにする。各サブクラスは、そのスーパークラスのフレーム及びメソッドを継承する。

【0006】オブジェクト指向プログラミングシステムは、一般に、同一アプリケーション内に同時に多数のアクティビティを発生させる「メツシーデスク」環境を実

現する。オブジェクト指向プログラミングシステムにおいては、カプセル化されたデータを含むオブジェクトに動作要求メッセージを送ることにより仕事は遂行される。オブジェクトは、当該オブジェクトに対して予め定義されているメソッドに従って、データに対して要求された動作を実行する。オブジェクトの「クラス」は、当該オブジェクトに与えられるデータ及び動作要求の「型」及び「意味」を定義する。データを含む各オブジェクトは各クラスの「インスタンス(個体)」と呼ばれる。要求された動作を実行するためのプログラムはクラスの「メソッド」と呼ばれる。

【0007】オブジェクトクラスは他のクラスのサブクラスとして定義されることもある。サブクラスは親クラスのすべてのデータ特性及びメソッドを継承する。サブクラスは付加的なデータ及びメソッドを加えることができると共に、さらに親クラスのどのデータ要求はメソッドでも再定義することができる。メツセージは、目標オブジェクトがそのメツセージを処理するためのメソッドを定義しているか又は当該メソッドを定義している親クラスをもっている限り、そのオブジェクトインスタンスのクラスのメソッド又はその親クラスのメソッドと、当該オブジェクトインスタンス内のデータとを使用して処理される。オブジェクト指向プログラミングシステムの継承特性は、本発明においては、システムティックな階層的パネル間プロセス制御機構の実現に依存する。

【0008】

【発明が解決しようとする課題】本発明の目的は1つのアプリケーションパネルから他のアプリケーションパネルに切り換えようというユーザの要求を処理するための、階層的集中制御管理を用いるパネル間プロセスフロー制御方法を提供することである。本発明の他の目的はオブジェクト指向プログラミングシステムの継承特性を利用するようにしたパネル間プロセスフロー制御方法を提供することである。本発明のさらに他の目的はパネル間の階層関係に依存するパネル間プロセスフロー制御のための、集中化されたシステムを提供することである。

【0009】

【課題を解決するための手段】かかる課題を解決するため本発明においては、集中化された制御管理手続きが、1つの事象として定義されるユーザ開始指令を受け取りかつ現在のアクティブパネルが上記開始指令を処理できるか否かを確認するようにし、これにより上記の目的及び他の目的並びに種々の利点が実現される。パネル化されたアプリケーション及びユーザ間の対話は、それぞれユーザがキーボード又はマウスを用いて事象を表現することによってなされる。各パネルによって処理し得る事象のリストが定義される。かかる事象の集合は、特定のパネルの「スコープ」を表現している。「スコープ」は、すなわち当該特定のパネルが応答し得る「ユーザが開始した要求」の集まりである。事象はアクティブ

パネルが処理できないとき、当該事象はスコープを変更するか又はアクティブパネルのスコープ外へ出る。本発明の最も重要な事項は、ユーザによる別のアプリケーションパネルの選択である。それは、キーボード入力又はマウスの移動とクリック及びタッチスクリーン入力とによって、表示されている他のパネルをアクティブアプリケーションパネルにすべくユーザが選択するとき生ずる。当該スコープ外にある命令をアクティブパネルが受け取ったとき、この命令はアクティブパネルの子パネルによって透過的に処理されるか、又はアクティブパネルの親パネルに戻されて処理される。多くの場合、各スコープ変更要求に対してパネルの木の部分木（サブツリー）が実行される。パネル間巡航の実際の順番は、現在のアクティブパネルのすぐ隣にある子パネルのうちの1つから開始する。この最初の子パネルが要求された事象を処理することができないとき、子パネルの子パネル達のうちの1つが試される。この処理は現在のアクティブパネルから出発して、葉パネル（すなわち後続パネルをもたない経路内パネル）を発見できるまで、この階層的経路を下り続ける。このようにしてユーザが開始した事象を処理することができるパネルを発見できるまでパネル間の各階層的経路が順次試される。ユーザ要求を処理できる子パネルがないことが判明したとき、当該要求は現在のアクティブパネルから親パネルに渡されて処理が継続される。

【0010】

【実施例】以下図面について本発明の一実施例を、オブジェクト指向環境との関連で説明する。図1、図2及び図3はそれぞれ、一般的オブジェクトクラス、オブジェクトクラスの一例及びオブジェクトの継承属性を表している。メソッド中にフレームを封じ込めてカプセル化したオブジェクトの概略的表現を図1に示す。

【0011】図2は、「エンプロイヘッダデータ入力パネル」と呼ばれるオブジェクトクラスを示し、多数のプログラムにより取り囲むように示されている。図2のプログラムは、入力受入れ機能、生成機能、初期設定機能、検索機能、処理機能及び表示機能を含む。例えば、生成機能はオブジェクトクラスエンプロイの新しいインスタンスを生成させ、入力受入れ機能はオブジェクトがキーボード又はマウス装置の入力からデータを受け入れることができるようにし、検索機能はオブジェクトクラスエンプロイの新しいインスタンスの中に収められているデータを検索する。

【0012】図3は「ユーザインタフェース」オブジェクトクラスについてのクラス継承階層を示す。このオブジェクトクラスは「子の属性リレー」を有し、すべてのサブクラスによって継承される「アクションルータ」メソッド及び「入力受入れ」メソッドを定義している。「アクションルータ」メソッドは、当該アクション要求が当該オブジェクトクラスによって処理されるかどうか

を確認する。「入力受入れ」メソッドはオブジェクトクラスが接続された入力機構からユーザ入力を受け入れることができるようにする。図3の「ユーザインタフェース」オブジェクト20は「パネル」サブクラス22及び「ポップアップ」サブクラス24の2つのサブクラスを有する。メソッドは「ユーザインタフェース」クラスから継承したメソッドを用いてこれらのサブクラスにより処理される。「パネル」クラス22の下位には「リストパネル」サブクラス26、「木パネル」サブクラス30、「アプリケーションリストパネル」34、「アプリケーション木パネル」36及び「アプリケーションパネル」28がある。「ポップアップ」クラス24の下位には、「アプリケーションポップアップ」32がある。上記各サブクラスは、「入力受入れ」メソッド及び「アクションルータ」メソッドを継承する。さらに、本発明に関連するものとして「処理」メソッドがある。「処理」メソッドへのインタフェースは「ユーザインタフェース」クラス20の中に定義され、このメソッドの実施は各子クラスに任せられる。

【0013】図4の概略的ブロックは具体化したオブジェクト指向コンピュータシステムの構成を表している。このシステムは各オブジェクトについての手続きを実行するデータプロセッサ11と、オブジェクト指向プログラム12と、使用中のデータ及び中間結果のための作業用記憶域を提供するランダムアクセスメモリ（RAM）13と、永久データを記憶する直接アクセス記憶装置（DASD）14と、各アプリケーションプログラムに割り当てられている1つ又は複数の画面を提供する表示用端末装置15とを含んでいる。

【0014】典型的なオブジェクト指向プログラム12は、各オブジェクトに関連するメソッドテーブルを含んでいる。このメソッドテーブルはメソッド番号と、メソッド番号に対応するメソッドが配設されているアドレスとを有する。通常各オブジェクトには、オブジェクトのすべてのインスタンス（個体）と、各インスタンスに対応するアドレス又は各インスタンスに関するオブジェクト参照（O REF）とを収容するオブジェクト識別テーブルが含まれている。これらのテーブルは、メソッドを実行する処理と、オブジェクト及びオブジェクトのデータインスタンスにアクセスする処理とに使用される。米国特許出願第425,607号及び第425,746号にオブジェクト指向プログラミングシステムについての一層詳細な記載がなされており、開示の技術を本発明の実施例に適用し得る。

【0015】オブジェクト指向プログラミングシステムにおいては、他のタスクに悪影響を与えることなく、多数のタスクを独立にかつ並列に処理することが望ましい。「メツシーデスク」環境においては、同一アプリケーション内に多数の並行アクティビティが存在する。

【0016】図5はパネル及びその子パネル間の階層関

係を示している。パネル1はシステムのプレゼンテーションマネージャパネル又は何等かのアクティブサブパネルで構成される。図において、パネル1は3つの子パネルすなわちパネル1.1、パネル1.2及びパネル1.3を有する。パネル1.1.1はパネル1.1の子パネルであり、パネル1.2.1及びパネル1.2.2はパネル1.2の子パネルであり、パネル1.3.1はパネル1.3の子パネルである。図の右側の矢印は、スコープ変更指令（すなわち、特定のパネルによつては処理し得ない事象）と、パネルからパネルに階層間に跨がっているデータフローとを示している。階層のプロセスを図解するため、いま仮にパネル1.2がアクティブパネルであり、またマウス又は表示画面カーソルをパネル1.2の位置からパネル1の可視領域中の点に移動させることによつてユーザがパネル1を選択するものとする。基礎的な制御プロセスにおいては、自分では処理できない事象に関するスコープ変更指令をパネル1.2が受け取ると、パネル1.2は当該スコープ変更情報を子パネル1.2.1に渡す。この場合パネル1.2.1は、そのスコープ変更指令を受け取ることができる子パネルを有していないので、パネル1.2.1は変更されていないデータ及びスコープ指令をその親パネルに返す。パネル1.2は次に、このスコープ変更指令をパネル1.2の他の子パネルに送る。パネル1.2.2はパネル1.2.1と同様に応答する。この時点で、パネル1.2はスコープ変更指令及びデータをその親パネル（パネル1）へ返す。パネル1はこのスコープ変更指令を処理できるので、パネル1がアクティブパネルとなる。

【0017】この実施例の階層的制御機構においては、各パネルがすべてのアクティブ子パネルのリストを保持しており、かつ継承された集中制御管理ルーチンを収容している。この集中制御管理ルーチンは、（A1）スコープ指令及びデータを入力パラメータとして受け取り、（A2）できれば当該スコープアクションそれ自身を処理し、（A3）できないならば当該スコープ指令を処理

するよう順次各子パネルに要請し、（A4）呼び出したオブジェクトにスコープ変更指令又は新しいアクションを返す。パネルは、そのパネル識別子がスコープ変更指令のパネル識別子に対する変更と同じであれば、スコープアクションそれ自身を処理することができる。次にこのパネルがアクティブパネルになり、他のスコープ変更が検出されるまでそれ自身に対するユーザアクションを処理する。このアクティブパネルは次にステップ（A2）を反復することによつて、上記4つのステップからなる制御ループプロセスを継続する。非アクティブパネルの集中制御管理ルーチンに入る時には、スコープ指令及びデータを入力パラメータとして受け取る第1のステップのみが生ずる。このルーチンはその後「アクションルータ」メソッドと呼ばれる。アクティブパネルは正規処理ロジックの一部としてスコープ変更指令を含んでいるすべてのユーザアクションを捕捉する。パネルがアクティブになるためには、次の操作のうちの1つが発生する必要がある。

【0018】（B1）親パネルが子パネルを生成して制御を引き渡す。

（B2）ユーザがパネルに対するスコープを変更する。

（B3）子パネルが処理を終了して親パネルにリターンする。

パネルフロッププロセスはオブジェクト指向クラス階層構造を利用して実行される。「パネル」クラスは子パネルの属性を有しており、この属性は現在のパネルのすべての子パネルについての連係リストである。この場合には、子パネルは継承属性を参照せず、その代わりに一次パネルから出る二次パネルを参照する。「パネル」クラスから継承する各パネルはこのリスト属性を継承する。「パネル」クラスは関連する3つのメソッドを有す。各メソッドのための擬似コードをこの明細書の別表に記載する。

【表1】

別表

1. アクションルータ

```
procedure action-router(  
    event : in out EVENT-TYPE;  
    did-i-do-it : in out BOOLEAN)  
<(* 経路制御 *)> is  
    end-of-list : BOOLEAN;  
begin  
    -- LOGIC  
loop  
    did-i-do-it := FALSE;  
    <(* アクションを処理すべく私のパネル操作ルーチンを呼び出す *)>  
    is  
begin  
        SELF.handle ( event, did-i-do-it );  
    end;  
    <(* 私のパネル操作ルーチンができたとき、次のユーザ応答を待つ。できな  
        かったとき、私に、彼らがその事象を処理できるかどうかを私の子にそれぞ  
        れ尋ねさせる。*)> is  
begin  
    if
```

(did-i-do-it = TRUE)
【0019】「アクションルータ」メソッドは集中制御管理ルーチンである。この集中制御管理ルーチンはスコープ変更指令を受け取って、当該スコープ変更指令がパネルそれ自身又は子パネルによつて処理され得るか否か、すなわちスコープ変更指令を当該子パネルの親に送り返す必要があるかを認める。これは階層的プロセスフローの中心要素である。パネルは当該パネル自身が処理し得る事象のリストに係わっているだけであるから、当該パネルのスコープ外にあるアクションは、当該パネルの子パネルによつて透過的に処理されるか又はそのパネルの親に送り返される。「パネル」クラスから継承する各パネルは「アクションルータ」メソッドを継承する。「処理」メソッドは現在のパネルに関連する指令を処理する。「パネル」クラスから委譲されているの

で、各パネルはそれ自身の「処理」メソッドを定義する。第3のメソッドは「入力受入れ」メソッドであり、このメソッドはマウス等が接続された入力装置又はキーボードからパネルが入力を受け取ることができるようにする。再び図3を参照しながら説明すれば、本発明の主要な概念は、「アクションルータ」メソッド及び「入力受入れ」メソッドが一度「ユーザインタフェース」20内に定義されると、次に「ユーザインタフェース」のすべての子パネルによつて継承されることである。「処理」メソッドは、特定の子パネルの「処理」メソッドが「アクションルータ」から呼び出されたとき「ユーザインタフェース」20の各子パネルによつて再定義される。

【0020】パネルオブジェクトクラス及び対応するメ

ツツの対話について、図5の階層的パネルレイアウトに関連させて説明する。出発点としてアクティブパネルとなっているパネル1、2を考える。これは、ユーザ入力をもっている「入力受入れ」メソッドの中にパネル1、2があることを意味する。ユーザが、例えばパネル1などの別のパネルを、パネル1、2の非重複領域からパネル1の非重複領域にカーソルをドラッグさせた後マウスボタンを押すことにより選択すると、スコープ変更を指示する事象が生成される。パネル1、2は「入力受入れ」メソッドから「アクションルータ」メソッドに戻す。「入力受入れ」メソッドはスコープ変更事象を戻す。パネル1、2の「アクションルータ」メソッドは当該事象の処理要求をパネル1、2の「処理」メソッドに送る。しかしながら、パネル1、2の「処理」メソッドはこのスコープ変更が当該パネルによって処理され得る事象のリスト外にあると判定する。「処理」メソッドはパネル1、2の「アクションルータ」メソッドに戻り、これが次にパネル1、2の「アクションルータ」メソッドを実行することによってスコープ変更情報をパネル1、2に渡す。實際上、パネル1、2の「アクションルータ」メソッドはパネル1、2の「アクションルータ」メソッドと厳密に同じであり、これは当該パネルによって当該パネルの親クラスから継承されたものである。パネル1、2の「アクションルータ」メソッドはスコープ変更事象を処理するためにパネル1、2の「処理」メソッドを要求する。パネル1、2の「処理」メソッドはこの事象を処理できないので、パネル1、2の「アクションルータ」メソッドに制御を戻す。パネル1、2の「アクションルータ」メソッドは次に事象スコープ変更を送りつけることができるような子パネルをパネル1、2がもっていないことを確認して、その親であるパネル1、2の「アクションルータ」メソッドに制御及び変更されていないスコープ事象を戻す。

【0021】パネル1、2はこの時点で、パネル1、2の「アクションルータ」メソッドにスコープ変更事象を送ることによって他の子パネルの検査を継続する。パネル1、2の「アクションルータ」メソッドは、スコープ変更事象を処理するために、パネル1、2の「処理」メソッドを要求する。この事象はパネル1、2の2つのスコープ外にあるので、「処理」メソッドはパネル1、2の「アクションルータ」メソッドに制御を戻す。パネル1、2の「アクションルータ」メソッドは当該パネルが子パネルをもっていないことを確認する。そこで親であるパネル1、2にスコープ変更事象が送り返され、スコープ事象が変更されないまままで制御がパネル1、2に戻される。

【0022】パネル1、2の「アクションルータ」メソッドは、スコープ変更事象を送りつけることができる子パネルが外には存在しないことを確認する。そして、こ

の事象を処理することができないので、その親であるパネル1にスコープ変更事象と共に制御を戻す。パネル1の「アクションルータ」メソッドは、その子パネルであるパネル1、2の「アクションルータ」メソッドから制御を受け取る。パネル1の「アクションルータ」メソッドは、スコープ変更事象を処理するためにパネル1の「処理」メソッドを要求する。最後にこの「処理」メソッドはこのスコープ変更事象を自身で処理し得ることを確認して、パネル1の「アクションルータ」メソッドに肯定的な応答を送る。パネル1の「アクションルータ」メソッドは次に、パネル1の「入力受入れ」メソッドを呼び出して、入力されるべき次のユーザ事象を待ち受ける。

【0023】(C1) アクションルータ
(C2) 処理
(C3) 入力受入れ

上述においては、特定の実施例に基づいて本発明を説明したが、本発明の精神及び範囲から離れることなく、その様式と詳細部分との双方に種々の変更を加えることができるものであり、またオブジェクト指向環境の中で動作するようにした実施例を特に取り上げて本発明を説明したが、本発明の方法は、同時に走る複数のアプリケーションのウインドウ表示を使用するマルチタスクコンピュータシステムのどのような様式のものに対しても適用することができる。

【0024】
【発明の効果】上述のように本発明によれば、オブジェクト指向プログラミング構成においてユーザによって作られたスコープ変更指令を、階層的内部プログラムロジックによって、継承属性を保持するように階層化されたパネルを選択制御することにより処理するようにしたことにより、プログラム及び手続きを一般と簡易化し得るデータ入力装置を容易に実現できる。

【図面の簡単な説明】

【図1】オブジェクトの概略的關係を示す略線図である。

【図2】オブジェクトの一例の概略的關係を示す略線図である。

【図3】「ユーザインタフェース」オブジェクトのクラス継承階層を示す系統図である。

【図4】本発明を実施するためのオブジェクト指向コンピュータシステムの概略的ブロック図である。

【図5】パネル及びその子パネル間の階層關係を示す系統図である。

【符号の説明】

10……オブジェクト指向コンピュータシステム、11……データプロセッサ、12……オブジェクト指向プログラム、13……ランダムアクセスメモリ(RAM)、14……直接アクセス記憶装置(DASD)、15……表示用端末装置、20……「ユーザインタフェース」オ

プロジェクト、2 2……「パネル」サブクラス、2 4……
「ポップアップ」サブクラス、2 6……「リストパネ
ル」サブクラス、2 8……アプリケーションパネル、3

0……「木パネル」サブクラス、3 2……アプリケーション
ポップアップ、3 4……アプリケーションリストパ
ネル、3 6……アプリケーション木パネル。

```

then
    SELF.Accept-Input (event);
else
    if
        (children.nb-elements>0)
    <(* 尋ねるべき子パネルが少なくとも1つは存在することを確かめる
        *) *> is
    then
        end-of-list := FALSE;
    <(* 連係リスト内の最初の子パネルに位置付ける *) *> is
    begin
        children.move-to-first;
    end;
    while
        ( ( did-i-do-it = FALSE)
          and
          ( end-of-list = FALSE))
    loop
    <(* この事象を処理できるかどうかをこの子に尋ねる *) *> is
    begin
        children.current-value.action-router
            (event, did-i-do-it );
    end;
    <(* もし、この子がそれをできなかつたならば、リスト内の次の
        子へと進む *) *> is
    begin
        if

```

```

        (did-i-do-it = FALSE)

    then

        children.move-to-next ( end-of-list );

    end if;

end;

end loop;

end if;

end if;

end;

<* もし、このパネルがこの事象を処理できず、しかもこのパネルの子
  の中にもこの事象を処理できるものが存在しないとき、この事象を
  このパネルの親パネルまで戻す *) > is

begin

    exit when ( did-i-do-it = FALSE );

end;

end loop;

end action-router;

```

II. 処理

```

procedure handle (

    event-in : in EVENT-TYPE;

    did-i-do-it : in out BOOLEAN)

<* (* コマンド処理する *) > is

begin

    --LOGIC

    <* (* ここではこの事象を処理できないものとみなして、フラグを偽にセ
      ヲトする *) > is

```

```

begin
    did-i-do-it := FALSE;
end;
<* (* 入ってくる事象を、このパネルが処理方法を知っているすべての事象
    と照合する *) *> is
begin
    if
        (event-in = "change to this panel's scope")
    then
        did-i-do-it := TRUE;
        SELF.change-to-this-panels-scope;
    elsif
        (event-in = "event-1")
    then
        did-i-do-it := TRUE;
        SELF.do-event-1;
    elsif
        (event-in = "event-2")
    then
        did-i-do-it := TRUE;
        SELF.do-event-2;
    end if;
end;
end handle;

```

Ⅲ. 入力受入れ

```

procedure accept-input (

```

```

event-out : out EVENT-TYPE)

<* (* ユーザが開始する事象を待受ける *) *> is
begin
  -- LOGIC

  <* (* ユーザ事象を待受けて受け取り、それをevent-out 内に格納する *)
    *> ;
end accept-input;

```

【図1】

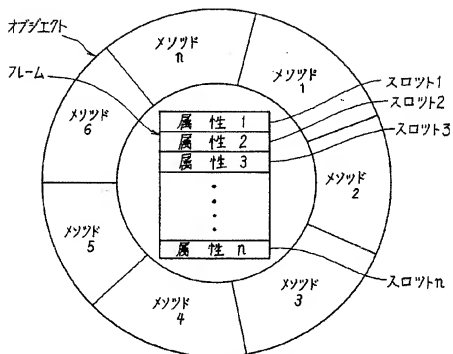


図1 オブジェクトの属性

【図2】

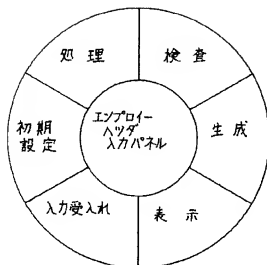


図2 オブジェクトクラスの内容

【図4】

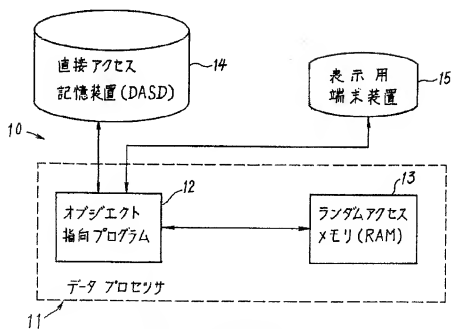


図4 オブジェクト指向コンピュータシステム

【図3】

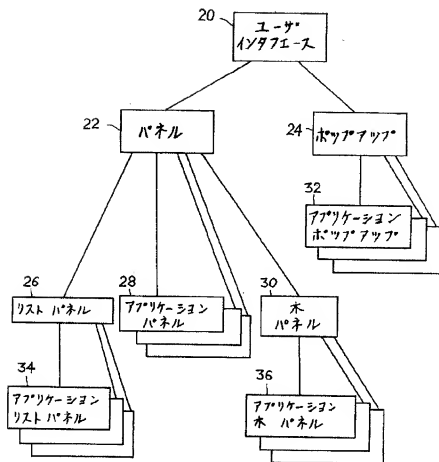


図3 クラス継承階層

【図5】

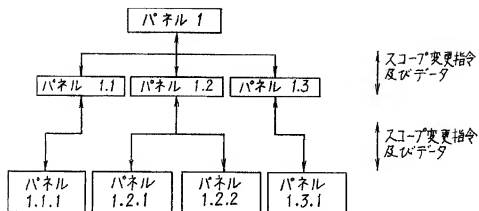


図5 パネル及び子パネル

フロントページの続き

(72) 発明者 リチャード・ユージーン・ムーア
アメリカ合衆国、ジョージア州30067、マ
リエッタ、プリンセス・レイン 2592番地